Finite Automata
Part One
Computability Theory
What problems can we solve with a computer?
What problems can we solve with a computer?

What is a “problem?”
Problems with Problems

• Before we can talk about what problems we can solve, we need a formal definition of a “problem.”

• We want a definition that
  • corresponds to the problems we want to solve,
  • captures a large class of problems, and
  • is mathematically simple to reason about.

• No one definition has all three properties.
Formal Language Theory
Strings

- An **alphabet** is a finite, nonempty set of symbols called **characters**.
  - Typically, we use the symbol $\Sigma$ to refer to an alphabet.
- A **string over an alphabet $\Sigma$** is a finite sequence of characters drawn from $\Sigma$.
- Example: If $\Sigma = \{a, b\}$, here are some valid strings over $\Sigma$:
  
  a   aabaaabbbabaaabaaaabbb   abbababba

- The **empty string** has no characters and is denoted $\varepsilon$.
- Calling attention to an earlier point: since all strings are finite sequences of characters from $\Sigma$, you cannot have a string of infinite length.
Languages

- A **formal language** is a set of strings.
- We say that $L$ is a **language over $\Sigma$** if it is a set of strings over $\Sigma$.
- Example: The language of palindromes over $\Sigma = \{a, b, c\}$ is the set
  - $\{\varepsilon, a, b, c, aa, bb, cc, aaa, aba, aca, bab, \ldots\}$
- The set of all strings composed from letters in $\Sigma$ is denoted $\Sigma^*$.
- Formally, we say that $L$ is a language over $\Sigma$ if $L \subseteq \Sigma^*$. 
How many of the following statements are true?

- **Alphabets** are sequences of characters.
- **Languages** are sets of strings.
- **Strings** are sets of characters.
- **Characters** are individual symbols.
- **Languages** are sequences of characters.

Answer at PollEv.com/cs103 or text CS103 to 22333 once to join, then a number.
To Recap

- **Languages** are sets of strings.
- **Strings** are sequences of characters.
- **Characters** are individual symbols.
- **Alphabets** are sets of characters.
Old MacDonald Had a Symbol, ♫ Σ-eye-ε-eyε, Oh! ♫

- You may have noticed that we have several letter-E-ish symbols in CS103, which can get confusing!
- Here’s a quick guide to remembering which is which:
  - In language theory, Σ refers to an *alphabet*.
  - In language theory, ε is the *empty string*, which is length 0.
  - In set theory, use ∈ to say “is an *element of*.”
  - In set theory, use ⊆ to say “is a *subset of*.”
What problems can we solve with a computer?
What problems can we solve with a computer?
What is a computer?

Bell Labs in Oakland, CA
(Photo by Larry Luckham, 1969)
Computers are Messy

That messiness makes it hard to rigorously say what we intuitively know to be true: that, on some fundamental level, different brands of computers or programming languages are more or less equivalent in what they are capable of doing.

\[ C \ vs \ C++ \ vs \ Java \ vs \ Python \]
We need a simpler way of discussing computing machines.
An *automaton* (plural: *automata*) is a mathematical model of a computing device.
Automata are Clean
Why Build Models?

- **Mathematical simplicity.**
  - It is significantly easier to manipulate our abstract models of computers than it is to manipulate actual computers.

- **Intellectual robustness.**
  - If we pick our models correctly, we can make broadly applicable claims about huge classes of real computers by arguing that they're just special cases of our more general models.
Two Models of a Machine

• The models of computation we will explore in this class correspond to different conceptions of what a computer could do.

• **Finite automata** (next two weeks) are an abstraction of computers with finite resource constraints.
  • Provide upper bounds for the computing machines that we can actually build.

• **Turing machines** (later) are an abstraction of computers with unbounded resources.
  • Provide upper bounds for what we could ever hope to accomplish.
What problems can we solve with a computer?
Finite Automata
A finite automaton is a simple type of mathematical machine for determining whether a string is contained within some language.
Each finite automaton consists of a set of states connected by transitions.
A Simple Finite Automaton

\[ q_0 \longrightarrow 0 \quad q_1 \]
\[ 1 \quad 1 \quad 1 \]
\[ q_3 \quad 0 \quad q_2 \]

start
A Simple Finite Automaton

Each circle represents a state of the automaton.
A Simple Finite Automaton
A Simple Finite Automaton

One special state is designated as the start state.
A Simple Finite Automaton
A Simple Finite Automaton
The automaton is run on an input string and answers “yes” or “no.”
A Simple Finite Automaton
A Simple Finite Automaton

$q_0$

$q_1$

$q_2$

$q_3$

start

0 1 0 1 1 1 0
A Simple Finite Automaton

The automaton can be in one state at a time. It begins in the start state.
A Simple Finite Automaton

\[ q_0 \xleftarrow{0} q_1 \xrightarrow{0} q_2 \xleftarrow{1} q_3 \xrightarrow{1} q_1 \]

Input sequence: 0 1 0 1 1 1 0
A Simple Finite Automaton
The automaton now begins processing characters in the order in which they appear.
A Simple Finite Automaton

\[ q_0 \quad q_1 \quad q_2 \quad q_3 \]

Start state: \( q_0 \)

Transitions:
- \( q_0 \) on 0 to \( q_1 \)
- \( q_0 \) on 1 to \( q_3 \)
- \( q_1 \) on 0 to \( q_0 \)
- \( q_1 \) on 1 to \( q_2 \)
- \( q_2 \) on 1 to \( q_1 \)
- \( q_3 \) on 1 to \( q_2 \)
- \( q_3 \) on 0 to \( q_0 \)

Input: 0 1 0 1 1 1 0
A Simple Finite Automaton

start

$q_0$ → $q_1$

$q_3$ → $q_2$

0 1 0 1 1 1 0
A Simple Finite Automaton

Each arrow in this diagram represents a transition. The automaton always follows the transition corresponding to the current symbol being read.
A Simple Finite Automaton

The diagram shows a finite automaton with the following states:
- $q_0$
- $q_1$
- $q_2$
- $q_3$

The transitions are:
- From $q_0$ to $q_1$ with input 0
- From $q_0$ to $q_3$ with input 1
- From $q_1$ to $q_2$ with input 0
- From $q_3$ to $q_2$ with input 0

The input sequence is: 0 1 0 1 1 1 0
A Simple Finite Automaton
A Simple Finite Automaton

start → \( q_0 \)

\( q_0 \) → \( q_1 \) on 0

\( q_1 \) → \( q_2 \) on 0

\( q_2 \) → \( q_3 \) on 0

\( q_3 \) → \( q_0 \) on 1

\( q_0 \) → \( q_3 \) on 1

\( q_1 \) → \( q_2 \) on 1

\( q_2 \) → \( q_1 \) on 1

\( q_3 \) → \( q_0 \) on 1

0 1 0 1 1 1 0
A Simple Finite Automaton

\begin{figure}[h]
\centering
\includegraphics[scale=0.8]{finite_automaton_diagram}
\end{figure}

The diagram represents a finite automaton with states \( q_0, q_1, q_2, q_3 \) and transitions labeled with 0s and 1s. The start state is \( q_0 \).
A Simple Finite Automaton

After transitioning, the automaton considers the next symbol in the input.
A Simple Finite Automaton

\[ q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \rightarrow q_0 \]

\[ 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \]
A Simple Finite Automaton

Diagram showing states $q_0$, $q_1$, $q_2$, and $q_3$ with transitions labeled '0' and '1'. The start state is $q_0$. The transition arrows indicate movements from one state to another based on input symbols '0' and '1'.
A Simple Finite Automaton

start

$q_0$ 0 $q_1$

1 1 1 1 1

$q_3$

$q_2$

0 0 0

0 1 0 1 1 1 0
A Simple Finite Automaton

The diagram represents a finite automaton with states $q_0$, $q_1$, $q_2$, and $q_3$. The transitions are labeled with input symbols 0 and 1. The start state is $q_0$. The automaton accepts the input string 0101110.
A Simple Finite Automaton

\[ q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \rightarrow q_0 \]

Start: \( q_0 \)

Transitions:
- \( 0 \to q_0 \)
- \( 0 \to q_1 \)
- \( 1 \to q_3 \)
- \( 0 \to q_2 \)
- \( 0 \to q_2 \)
- \( 1 \to q_3 \)
- \( 1 \to q_3 \)
- \( 1 \to q_3 \)
- \( 1 \to q_3 \)

Input: 0101110
A Simple Finite Automaton

- Start state: $q_0$
- Transitions:
  - $q_0 \xrightarrow{0} q_1$
  - $q_0 \xrightarrow{1} q_3$
  - $q_1 \xrightarrow{0} q_0$
  - $q_1 \xrightarrow{0} q_2$
  - $q_3 \xrightarrow{1} q_1$
  - $q_3 \xrightarrow{1} q_2$
  - $q_2 \xrightarrow{0} q_3$

Input sequence: 0 1 0 1 1 1 0
A Simple Finite Automaton

start

$q_0$ 0

1 1

$q_3$ 0

1

$q_2$ 1 1 1

$q_1$ 0
A Simple Finite Automaton
A Simple Finite Automaton
A Simple Finite Automaton

\[
\begin{array}{c}
\text{start} \\
q_0 & \xrightarrow{0} & q_1 \\
\uparrow & & \uparrow \\
q_3 & \xrightarrow{1} & q_2 \\
& & \\
\end{array}
\]

Input: 0 1 0 1 1 1 0
A Simple Finite Automaton
A Simple Finite Automaton
A Simple Finite Automaton

\[
\begin{array}{c}
\text{start} \\
q_0 \\
\downarrow \quad 1 \quad 1 \\
q_3 \\
\quad \downarrow \quad 0 \\
q_2 \\
\quad \downarrow \quad 0 \\
q_1 \\
\end{array}
\]

Input sequence: 0 1 0 1 1 1 0
A Simple Finite Automaton

The diagram shows a finite automaton with the following states and transitions:

- **States:** $q_0$, $q_1$, $q_2$, $q_3$
- **Start State:** $q_0$
- **Transitions:**
  - From $q_0$ to $q_1$: on input 0
  - From $q_0$ to $q_3$: on input 1
  - From $q_1$ to $q_2$: on input 1
  - From $q_2$ to $q_3$: on input 1

The input string $0101110$ is processed through the automaton, starting at state $q_0$. The automaton transitions through states $q_0$, $q_1$, $q_2$, and finally reaches an accepting state $q_3$.
A Simple Finite Automaton

- Start state: $q_0$
- States: $q_0, q_1, q_2, q_3$
- Transitions:
  - $q_0 \rightarrow q_1$ on input 0
  - $q_0 \rightarrow q_3$ on input 1
  - $q_1 \rightarrow q_2$ on input 0
  - $q_3 \rightarrow q_0$ on input 1

Input sequence: 0 1 0 1 1 1 0
A Simple Finite Automaton
A Simple Finite Automaton
A Simple Finite Automaton

\[ q_0 \xrightarrow{0} q_1 \]

\[ q_3 \xrightarrow{1} q_2 \]

\[ q_2 \xrightarrow{0} q_3 \]

\[ q_1 \xrightarrow{0} q_0 \]

Input: 0 1 0 1 1 1 0
A Simple Finite Automaton

```
0 1 0 1 1 1 0
```
A Simple Finite Automaton
A Simple Finite Automaton
A Simple Finite Automaton

Now that the automaton has looked at all this input, it can decide whether to say "yes" or "no."
A Simple Finite Automaton

Now that the automaton has looked at all this input, it can decide whether to say “yes” or “no.”

The double circle indicates that this state is an accepting state, so the automaton outputs “yes.”
Now that the automaton has looked at all this input, it can decide whether to say "yes" or "no."

The double circle indicates that this state is an **accepting state**, so the automaton outputs "yes."

Now that the automaton has looked at all this input, it can decide whether to say "yes" or "no."
A Simple Finite Automaton

Now that the automaton has looked at all this input, it can decide whether to say "yes" or "no."

The double circle indicates that this state is an **accepting state**, so the automaton outputs "yes."

Now that the automaton has looked at all this input, it can decide whether to say "yes" or "no."
A Simple Finite Automaton
A Simple Finite Automaton

start

$q_0$  

$q_1$  

$q_2$  

$q_3$

1 1 1 1 1 1 0

1 0 1 0 0 0
A Simple Finite Automaton
A Simple Finite Automaton

The diagram shows a finite automaton with states $q_0$, $q_1$, $q_2$, and $q_3$. The transitions are labeled with 0 and 1. The start state is $q_0$. The input string is 10110000.
A Simple Finite Automaton
A Simple Finite Automaton

\[ q_0 \rightarrow 0 \rightarrow q_1 \]
A Simple Finite Automaton

\[
\begin{array}{c}
\text{start} \\
q_0 \\
q_3 \\
q_2 \\
q_1
\end{array}
\]

Transitions:
- From \( q_0 \):
  - \( 0 \) to \( q_1 \)
  - \( 1 \) to \( q_3 \)
- From \( q_1 \):
  - \( 0 \) to \( q_2 \)
  - \( 1 \) to \( q_0 \)
- From \( q_2 \):
  - \( 0 \) to \( q_3 \)
- From \( q_3 \):
  - \( 0 \) to \( q_0 \)

Input string: 1 0 1 0 0 0 0
A Simple Finite Automaton
A Simple Finite Automaton

start

$q_0 \rightarrow 0 \rightarrow q_1$

$1 \rightarrow q_3 \rightarrow 0 \rightarrow q_2$

$0 \rightarrow 1 \rightarrow 1 \rightarrow 0 \rightarrow 1 \rightarrow 1$

$1 0 1 0 0 0 0$
A Simple Finite Automaton

![Finite Automaton Diagram]

The diagram illustrates a finite automaton with states $q_0$, $q_1$, $q_2$, and $q_3$. The transitions are labeled with inputs 0 and 1, indicating the acceptance of strings composed of 0s and 1s.
A Simple Finite Automaton

Start state: $q_0$

States:
- $q_0$
- $q_1$
- $q_2$
- $q_3$

Transitions:
- From $q_0$ to $q_1$: input 0
- From $q_1$ to $q_0$: input 0
- From $q_0$ to $q_3$: input 1
- From $q_3$ to $q_0$: input 1
- From $q_3$ to $q_2$: input 1
- From $q_2$ to $q_3$: input 0
- From $q_2$ to $q_1$: input 0

Input sequence: 1 0 1 0 0 0 0
A Simple Finite Automaton
A Simple Finite Automaton

\[
\begin{array}{c}
q_0 \\
\downarrow 1 \\
q_3 \\
\downarrow 1 \\
\downarrow 1 \\
q_2 \\
\uparrow 1 \\
q_1 \\
\uparrow 0 \\
q_0
\end{array}
\]
A Simple Finite Automaton
A Simple Finite Automaton
A Simple Finite Automaton

![Diagram of a simple finite automaton with states q₀, q₁, q₂, q₃ and transitions labeled with 0s and 1s.]
A Simple Finite Automaton

\[ q_0 \quad \arrow{0} \quad q_1 \]

\[ q_3 \quad \arrow{1} \quad q_2 \]

\[
\begin{array}{c}
\text{start} \\
q_0 \quad \arrow{0} \quad q_1 \\
q_3 \quad \arrow{1} \quad q_2 \\
q_2 \quad \arrow{0} \quad q_3 \\
\end{array}
\]

\[
\begin{array}{cccccc}
1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 \\
\end{array}
\]
A Simple Finite Automaton
A Simple Finite Automaton
A Simple Finite Automaton

\[ q_0 \xrightarrow{1} q_3 \xrightarrow{1} q_2 \xrightarrow{1} q_1 \xrightarrow{0} q_0 \]
A Simple Finite Automaton
A Simple Finite Automaton

![Diagram of a finite automaton with states q0, q1, q2, q3, and transitions on 0 and 1]

Input: 1 0 1 0 0 0
A Simple Finite Automaton

\[
\begin{array}{c}
\text{start} \\
q_0 \rightarrow \q_3 \rightarrow q_2 \rightarrow \q_1 \rightarrow q_0
\end{array}
\]
A Simple Finite Automaton

start

$q_0$ 0 $q_1$

1 1 1 1 1

$q_3$ 0 $q_2$

1 0 1 0 0 0
A Simple Finite Automaton

![Finite Automaton Diagram]

- Start state: $q_0$
- States: $q_0, q_1, q_2, q_3$
- Transitions:
  - $q_0$ to $q_1$ on input 0
  - $q_1$ to $q_2$ on input 1
  - $q_2$ to $q_3$ on input 1
  - $q_3$ to $q_1$ on input 1
  - $q_1$ to $q_2$ on input 0

Input sequence: 1 0 1 0 0 0 0
A Simple Finite Automaton

[Diagram of a finite automaton with states $q_0$, $q_1$, $q_2$, $q_3$, and transitions labeled with 0s and 1s.]

Input: 1 0 1 0 0 0
A Simple Finite Automaton

\begin{center}
\begin{tikzpicture}[node distance = 2cm, on grid, auto, minimum size=30pt]
  \node[state,fill=yellow] (q0) {$q_0$};
  \node[state] (q1) [right of=q0] {$q_1$};
  \node[state] (q3) [below of=q0] {$q_3$};
  \node[state] (q2) [right of=q3] {$q_2$};

  \path[->,thick]
  (q0) edge [left] node {start} (q0)
  (q0) edge [below] node {1} (q3)
  (q0) edge [above] node {1} (q1)
  (q1) edge [below] node {0} (q2)
  (q1) edge [above] node {0} (q0)
  (q3) edge [below] node {0} (q2)
  (q3) edge [above] node {1} (q0)
  (q2) edge [below] node {1} (q1)
  (q2) edge [above] node {1} (q3);
\end{tikzpicture}
\end{center}

```
1 0 1 0 0 0
```
A Simple Finite Automaton
A Simple Finite Automaton

This state is not an accepting state (it’s a rejecting state), so the automaton says “no.”
A Simple Finite Automaton

This state is not an accepting state (it’s a **rejecting state**), so the automaton says “no.”
A Simple Finite Automaton

This state is not an accepting state (it’s a rejecting state), so the automaton says “no.”
A Simple Finite Automaton

Start

$q_0$ → $q_1$

$q_0$ → $q_3$

$q_3$ → $q_2$

$q_1$ → $q_2$

$q_2$ → $q_0$

$q_3$ → $q_1$

$q_2$ → $q_3$
A Simple Finite Automaton

Try it yourself! Does this automaton accept (vote A) or reject (vote R)?

Answer at PollEv.com/cs103 or text CS103 to 22333 once to join, then A or R.
The Story So Far

- A **finite automaton** is a collection of **states** joined by **transitions**.
- Some state is designated as the **start state**.
- Some states are designated as **accepting states**.
- The automaton processes a string by beginning in the start state and following the indicated transitions.
- If the automaton ends in an accepting state, it **accepts** the input.
- Otherwise, the automaton **rejects** the input.
Just Passing Through
Just Passing Through

\[\text{start} \rightarrow q_0 \]

\[
\begin{array}{c}
q_0 \quad 0 \\
q_1 \quad 1 \\
q_2 \quad 0 \\
q_3 \quad 1 \\
q_4 \quad 0 \\
\end{array}
\]

\[
\begin{array}{c}
1 \\
0 \\
1 \\
0 \\
1 \\
\end{array}
\]
Just Passing Through

- From start to $q_0$, $1 \rightarrow 0$
- From $q_0$ to $q_1$, $1 \rightarrow 0$
- From $q_0$ to $q_2$, $1 \rightarrow 0$
- From $q_1$ to $q_2$, $1 \rightarrow 1$
- From $q_2$ to $q_1$, $1 \rightarrow 0$
- From $q_2$ to $q_3$, $1 \rightarrow 0$
- From $q_2$ to $q_4$, $1 \rightarrow 0$
- From $q_3$ to $q_1$, $0 \rightarrow 1$
- From $q_3$ to $q_4$, $0 \rightarrow 1$
- From $q_4$ to $q_2$, $0 \rightarrow 1$
- From $q_4$ to $q_3$, $0 \rightarrow 1$

Input sequence: $1101$
Just Passing Through

Diagram:

- Start state: $q_0$
- States: $q_0, q_1, q_2, q_3, q_4$
- Edges:
  - $q_0$ to $q_1$: 1
  - $q_0$ to $q_2$: 0
  - $q_1$ to $q_0$: 0
  - $q_1$ to $q_3$: 1
  - $q_1$ to $q_4$: 1
  - $q_2$ to $q_0$: 1
  - $q_2$ to $q_1$: 1
  - $q_3$ to $q_1$: 1
  - $q_3$ to $q_4$: 0
  - $q_4$ to $q_1$: 0
  - $q_4$ to $q_3$: 0

Input: 1 1 0 1
Just Passing Through

(start \rightarrow q_0)

$q_0 \rightarrow q_1, q_2$ (0)

$q_1 \rightarrow q_2, q_3$ (1)

$q_2 \rightarrow q_4$ (1)

$q_3 \rightarrow q_4$ (0)

$q_4 \rightarrow q_0$ (1)

Input: 1 1 0 1
Just Passing Through

\[
\begin{array}{c}
\begin{array}{c}
\text{start} \\
1 \rightarrow q_0 \\
0 \\
\end{array} \\
\begin{array}{c}
q_1 \\
1 \rightarrow q_3 \\
1 \\
0 \\
\end{array} \\
\begin{array}{c}
q_2 \\
1 \rightarrow q_4 \\
1 \\
0 \\
\end{array} \\
\begin{array}{c}
q_3 \\
0 \\
1 \\
0 \\
\end{array} \\
\begin{array}{c}
q_4 \\
0 \\
1 \\
0 \\
\end{array}
\end{array}
\]

1 1 0 1
Just Passing Through

1 1 1 0 1
Just Passing Through

\[
\begin{array}{c}
\text{start} \\
qu_0 \\
qu_1 \\
qu_2 \\
qu_3 \\
qu_4 \\
\end{array}
\]

1 1 0 1
Just Passing Through

![Diagram of a finite state machine with states $q_0$, $q_1$, $q_2$, $q_3$, $q_4$ and transitions labeled with 0 and 1. The sequence 1101 is shown moving through the machine from start to $q_2$.](attachment:diagram.png)
Just Passing Through

\[ \begin{array}{c}
\text{start} \\
q_0 \\
\text{1} \\
\text{0} \\
q_1 \\
\text{0} \\
\text{1} \\
1 \\
q_2 \\
\text{1} \\
\text{0} \\
q_3 \\
\text{0} \\
\text{1} \\
1 \\
q_4 \\
\text{0} \\
\text{1} \\
1 \\
\end{array} \]
Just Passing Through

![Transition Diagram]

1 1 0 1
Just Passing Through

Graph representation:

- Start state: $q_0$
- States: $q_1, q_2, q_3, q_4$
- Edges:
  - $q_0$ to $q_1$: 1
  - $q_0$ to $q_2$: 0
  - $q_1$ to $q_3$: 1
  - $q_1$ to $q_4$: 0
  - $q_2$ to $q_3$: 1
  - $q_2$ to $q_4$: 0

Sequence: 1 1 0 1
Just Passing Through

![Finite automaton diagram with states: start -> q_0 -> q_1 -> q_2 -> q_3 -> q_4. Edges labeled with 0, 1, and transitions for 1101 sequence shown in red.]
Just Passing Through

start

$q_0$ 1 0

$q_1$ 1

$q_2$ 0 1

$q_3$ 1 0 1

$q_4$ 1 0

$1 1 0 1$
A finite automaton does not accept as soon as it enters an accepting state.

A finite automaton accepts if it ends in an accepting state.
What Does This Accept?
What Does This Accept?
What Does This Accept?

---

![Diagram of a finite automaton with states q0, q1, q2, q3, and q4. The initial state is q0, and the transitions are as follows:

- From q0 to q1 on input 1.
- From q0 to q2 on input 0.
- From q1 to q2 on input 0.
- From q1 to q3 on input 1.
- From q1 to q4 on input 0.
- From q2 to q1 on input 0.
- From q2 to q4 on input 0.
- From q3 to q1 on input 1.
- From q3 to q2 on input 0.
- From q3 to q3 on input 0.
- From q4 to q4 on input 0.

The states q1, q2, q3, and q4 are marked as accepting states.]}
What Does This Accept?
What Does This Accept?

- $q_0$: Start state
- $q_1$, $q_2$, $q_3$, $q_4$: States
- Transitions: 1 and 0
What Does This Accept?

The diagram represents a finite automaton with states $q_0$, $q_1$, $q_2$, $q_3$, and $q_4$. The transitions are labeled with symbols 0 and 1. The start state is $q_0$.
What Does This Accept?
No matter where we start in the automaton, after seeing two 1's, we end up in accepting state $q_3$. 
What Does This Accept?
What Does This Accept?

The diagram shows a finite automaton with states $q_0, q_1, q_2, q_3, q_4$. The start state is $q_0$. Transitions are labeled with input symbols:

- From $q_0$ to $q_1$: on input 0.
- From $q_0$ to $q_2$: on input 1.
- From $q_1$ to $q_0$: on input 1.
- From $q_1$ to $q_3$: on input 1.
- From $q_1$ to $q_4$: on input 0.
- From $q_2$ to $q_0$: on input 1.
- From $q_2$ to $q_4$: on input 0.
- From $q_3$ to $q_0$: on input 0.
- From $q_3$ to $q_4$: on input 0.
- From $q_4$ to $q_2$: on input 0.

Each state has a transition back to itself on either 0 or 1, as indicated by the loops.

The automaton accepts strings based on the path taken from the start state $q_0$.
What Does This Accept?
What Does This Accept?

The diagram depicts a finite automaton with the following states:

- Start state: $q_0$
- States: $q_0, q_1, q_2, q_3, q_4$
- Transitions:
  - From $q_0$ to $q_1$: 1
  - From $q_0$ to $q_2$: 0
  - From $q_1$ to $q_0$: 0
  - From $q_1$ to $q_2$: 1
  - From $q_2$ to $q_3$: 1
  - From $q_2$ to $q_4$: 0
  - From $q_3$ to $q_0$: 1
  - From $q_3$ to $q_1$: 0
  - From $q_4$ to $q_0$: 0

The automaton accepts strings that end with 0.
What Does This Accept?
What Does This Accept?

![Diagram of a finite automaton with states q0, q1, q2, q3, and q4, labeled with transitions for 0 and 1. The diagram starts at q0 and includes loops and arrows indicating transitions between states.](image_url)
No matter where we start in the automaton, after seeing two 0's, we end up in accepting state $q_5$. 
What Does This Accept?

*Diagram of a finite automaton with states $q_0, q_1, q_2, q_3, q_4$ and transitions on input symbols 0 and 1.*
What Does This Accept?

This automaton accepts a string in \( \{0, 1\}^* \) iff the string ends in \( 00 \) or \( 11 \).
The *language of an automaton* is the set of strings that it accepts.

If $D$ is an automaton that processes characters from the alphabet $\Sigma$, then $\mathcal{L}(D)$ is formally defined as

$$\mathcal{L}(D) = \{ w \in \Sigma^* \mid D \text{ accepts } w \}$$
How many of the following statements are true?

- A *language* of an automaton can have an infinitely long string (or many of them) in it.
- A *language* of an automaton can contain infinitely many strings.
- A *language* of an automaton can contain no strings.

Answer at PollEv.com/cs103 or text CS103 to 22333 once to join, then a number.
A Small Problem
A Small Problem

A finite automaton with states $q_0$, $q_1$, $q_2$ and transitions labeled with 0 and 1.
A Small Problem
A Small Problem

```
\begin{align*}
\text{start} & \rightarrow q_0 \\
0 & \rightarrow q_0 \\
0 & \rightarrow q_2 \\
1 & \rightarrow q_1 \\
0 & \rightarrow q_1 \\
1 & \rightarrow q_1 \\
1 & \rightarrow q_1 \\
0 & \rightarrow q_1 \\
\end{align*}
```
A Small Problem

The diagram shows a state transition diagram with states $q_0$, $q_1$, and $q_2$. The states are connected by transitions labeled with 0 and 1.

- The start state is $q_0$.
- From $q_0$, there is a transition on 0 to $q_2$.
- From $q_0$, there is a transition on 0 to $q_1$.
- From $q_1$, there is a transition on 1 to $q_2$.

The input sequence is 01100.
A Small Problem
A Small Problem

![Diagram of a finite automaton with states $q_0$, $q_1$, and $q_2$. $q_0$ is the start state, transitions are indicated by arrows with labels 0 and 1, and there is a sequence of inputs 01110 shown moving from $q_0$ to $q_1$.]
A Small Problem

\[
\begin{align*}
q_2 & \rightarrow q_0 \quad \text{(start)} \\
q_0 & \rightarrow q_0 (0) \\
q_0 & \rightarrow q_1 (1) \\
q_1 & \rightarrow q_0 (0) \\
q_1 & \rightarrow q_1 (1) \\
q_2 & \rightarrow q_2 (1)
\end{align*}
\]
A Small Problem
Another Small Problem

\[
\begin{align*}
q_0 & \xrightarrow{0, 1} q_1 \\
\text{start} & \xrightarrow{0, 1} q_0 \\
q_1 & \xrightarrow{0} q_2 \\
q_2 & \xrightarrow{0, 1} q_1
\end{align*}
\]
Another Small Problem

\[ q_0 \xrightarrow{0, 1} q_1 \xrightarrow{0, 1} q_2 \xrightarrow{0} \]
Another Small Problem

The diagram shows a finite automaton with states $q_0$, $q_1$, and $q_2$. The transitions are as follows:

- From $q_0$, on input 0,1, move to $q_1$.
- From $q_1$, on input 0,1, move to $q_2$.
- From $q_2$, on input 0, move to $q_1$.

The initial state is $q_0$ and it is marked as 'start'. The accept state is $q_2$. The input sequence shown is '0000' and it is indicated by the green arrow pointing up.
Another Small Problem

Diagram:

- Start state $q_0$
- Transition from $q_0$ to $q_1$ on inputs 0, 1
- Transition from $q_1$ to $q_2$ on inputs 0, 1
- Transition from $q_2$ to $q_0$ on inputs 0, 1

Input sequence: 0 0 0 0
Another Small Problem
Another Small Problem

\[
\begin{array}{c}
\text{start} \\
q_0 \quad 0, 1 \quad q_1 \quad 0, 1 \quad q_2 \\
0, 1 \\
\end{array}
\]
Another Small Problem

\[ q_0 \rightarrow q_1 \rightarrow q_2 \]

0, 1

\[ 0, 1 \]

0

0

Start

I HAVE NO IDEA WHAT I'M DOING
The Need for Formalism

• In order to reason about the limits of what finite automata can and cannot do, we need to formally specify their behavior in all cases.

• All of the following need to be defined or disallowed:
  • What happens if there is no transition out of a state on some input?
  • What happens if there are multiple transitions out of a state on some input?
DFAs

- A **DFA** is a
  - **D**eterministic
  - **F**inite
  - **A**utomaton
- DFAs are the simplest type of automaton that we will see in this course.
DFAs

• A DFA is defined relative to some alphabet Σ.

• For each state in the DFA, there must be exactly one transition defined for each symbol in Σ.
  • This is the “deterministic” part of DFA.

• There is a unique start state.

• There are zero or more accepting states.
How many of these are valid DFAs over \{0, 1\}? 

Answer at PollEv.com/cs103 or text CS103 to 22333 once to join, then a number.
Is this a DFA?
Designing DFAs

- At each point in its execution, the DFA can only remember what state it is in.

**DFA Design Tip:** Build each state to correspond to some piece of information you need to remember.

- Each state acts as a “memento” of what you're supposed to do next.
- Only finitely many different states means only finitely many different things the machine can remember.
Recognizing Languages with DFAs

\[ L = \{ w \in \{a, b\}^* | \text{the number of } b \text{'s in } w \text{ is congruent to two modulo three} \} \]
Recognizing Languages with DFAs

$L = \{ w \in \{a, b\}^* | \text{the number of } b\text{'s in } w \text{ is congruent to two modulo three} \}$
Recognizing Languages with DFAs

$L = \{ \ w \in \{a, b\}^*| \ \text{the number of } b's \text{ in } w \text{ is congruent to two modulo three} \ \}$
Recognizing Languages with DFAs

\[ L = \{ w \in \{a, b\}^* \mid \text{the number of b's in } w \text{ is congruent to two modulo three}\} \]
Recognizing Languages with DFAs

$L = \{ w \in \{a, b\}^* \mid \text{the number of } b\text{'s in } w \text{ is congruent to two modulo three} \}$
Recognizing Languages with DFAs

$L = \{ w \in \{a, b\}^* | \text{the number of } b's \text{ in } w \text{ is congruent to two modulo three} \}$
Recognizing Languages with DFAs

\[ L = \{ w \in \{a, b\}^* | \text{the number of } b's \text{ in } w \text{ is congruent to two modulo three} \} \]
Recognizing Languages with DFAs

$L = \{ w \in \{a, b\}^* | \text{the number of } b\text{'s in } w \text{ is congruent to two modulo three } \}$
Recognizing Languages with DFAs

$L = \{ w \in \{a, b\}^* \mid \text{the number of } b's \text{ in } w \text{ is congruent to two modulo three} \}$
Recognizing Languages with DFAs

\[ L = \{ w \in \{a, b\}^* \mid \text{the number of b's in } w \text{ is congruent to two modulo three} \} \]
Recognizing Languages with DFAs

$L = \{ \ w \in \{a, b\}^* \mid w \text{ contains aa as a substring} \ \}$
Recognizing Languages with DFAs

\[ L = \{ w \in \{a, b\}^* | w \text{ contains } aa \text{ as a substring} \} \]
Recognizing Languages with DFAs

$L = \{ w \in \{a, b\}^* \mid w \text{ contains } \text{aa} \text{ as a substring} \}$
Recognizing Languages with DFAs

$L = \{ w \in \{a, b\}^* \mid w\text{ contains } aa\text{ as a substring}\}$
Recognizing Languages with DFAs

\[ L = \{ w \in \{a, b\}^* | w \text{ contains } aa \text{ as a substring } \} \]
Recognizing Languages with DFAs

\[ L = \{ w \in \{a, b\}^* \mid w \text{ contains } aa \text{ as a substring } \} \]
Recognizing Languages with DFAs

\[ L = \{ w \in \{a, b\}^* \mid w \text{ contains } aa \text{ as a substring} \} \]
Recognizing Languages with DFAs

\[ L = \{ \ w \in \{a, b\}^* \mid \text{w contains } aa \text{ as a substring} \ \} \]
Recognizing Languages with DFAs

$L = \{ \ w \in \{ a, b \}^* \mid w \text{ contains } aa \text{ as a substring } \}$
Recognizing Languages with DFAs

\[ L = \{ w \in \{a, b\}^* \mid w \text{ contains } aa \text{ as a substring} \} \]
Recognizing Languages with DFAs

\[ L = \{ w \in \{a, b\}^* \mid w \text{ contains aa as a substring} \} \]
More Elaborate DFAs

$L = \{ \ w \in \{ a, *, / \}^* \mid w \text{ represents a C-style comment} \ \}$

Let’s have the $a$ symbol be a placeholder for “some character that isn’t a star or slash.”

Try designing a DFA for comments! Here’s some test cases to help you check your work:

Accepted:

```plaintext
/*a*/
/**/
/***/
/*aaa*aaa*/
/*a/a*/
```

Rejected:

```plaintext
/**
/**/a/*aa*/
aaa/**/aa
/**a/
/**a/a/a/a
```
More Elaborate DFAs

$L = \{ w \in \{ a, *, / \}^* \mid w \text{ represents a C-style comment} \}$